Table-Driven Configuration and Formatting of Telemetry Data in the Deep Space Network

Evan Manning

Telos Information Systems
320 N Halstead, Suite 260
Pasadena, CA 91107
manning@alumni.caltech.edu

Abstract

With a restructured software architecture for telemetry system control and data processing, the NASA/Deep Space Network (DSN) has substantially improved its ability to accommodate a wide variety of spacecraft in an era of "better, faster, cheaper."

In the new architecture, the permanent software implements all capabilities needed by any system user, and text tables specify how these capabilities are to be used for each spacecraft. Most changes can now be made rapidly, outside of the traditional software development cycle. The system can be updated to support a new spacecraft through table changes rather than software changes, reducing the implementation-test-and-delivery cycle for such a change from three months to three weeks. The mechanical separation of the text table files from the program software, with tables only loaded into memory when that mission is being supported, dramatically reduces the level of regression testing required.

The format of each table is a different compromise between ease of human interpretation, efficiency of computer interpretation, and flexibility.

1. Deep Space Network Telemetry 1990

In 1990 NASA's Deep Space Network (DSN) supported fewer than a dozen spacecraft, all them using minor variants on a single NASA output format. Each new spacecraft was a major event, frequently accompanied by upgrades of DSN hardware and software.

In addition, support for frequent minor processing changes was creating a bottleneck because each change required a formal software build and delivery, and regression testing. One example of such a change is an increase in data rate and frame size as an encounter approaches.

2. Changing Environment

In recent years both the number and variety of missions supported by DSN has grown explosively. Today, the DSN supports over seventy deep-space and near-Earth spacecraft operated by NASA/JPL, other NASA centers (e.g., GSFC), other US government agencies (e.g., NOAA), and other nations' space agencies (e.g., CNES, ISAS). And these spacecraft are beginning to be produced with shorter lead times.

It would be impossible to support all these spacecraft with the old system.

3. Tables Save the Day

Spacecraft specifics had to be removed from the main build-test-delivery cycle.

Text tables presented an opportunity to isolate mission-specifics from the telemetry processing software, and thus from much of the delivery cycle's cost in time and money. With a software architecture where tables are clearly read in by the computer only when the appropriate mission is commanded, table changes need no software build and little regression testing.

Three tables are sufficient to encapsulate all mission specific behaviors of telemetry processing: spacecraft initialization tables, rules tables, and format tables.

3.1 Spacecraft Initialization Tables

The spacecraft initialization table (SIT) configures devices based on mission-specific telemetry parameters (subcarrier frequency, coding type, frame length, Reed-Solomon interleave depth, etc.). Their format is almost exactly the same as that used for interactive Operator Directives (ODs) except for the addition of comments. Implementation of these tables was integrated with implementation of a warmstart file capability. Both send commands to the existing front end for interpretation, and so are easy to implement despite their power.

Example 1 is a SIT table.

3.1.1 Tradeoffs in SIT Table Design

SIT table design is natural, giving ease of operator use through familiarity and ease of implementation through use of existing interpretation facilities. The only loss in going to a table driven approach is a loss in speed because each command must be interpreted each spacecraft tracking pass.

3.2 Rules Tables

Realtime changes of certain key configuration parameters sometimes require changes to other related configuration parameters. For example, a change of bit rate may imply changes to frame length, coding type, and data output format.

Rules tables reconfigure devices and data output formats when key parameters change. The current implementation can accept only data rate as the key parameter because that is the only key parameter needed by any existing mission for changes to anything but format.

3.2.1 Rules Tables Format

The format of rules tables also uses the existing operator directive format as much as possible. The only enhancement is that these tables have two columns of ODs: key parameters in the first column, corresponding ODs to be processed in the second column. Although the first column is identical in format to the telemetry bit rate (TBR) OD, its meaning in context is different. When the operator enters "TBR 32000" the TGC reacts by configuring hardware to expect incoming data to change data rate to 32,000 bits per second (including any commands specified in the rules table). "TBR 32000" in the first column of a rules table directs the TGC to execute the corresponding ODs whenever a TBR OD is received with a new bit rate closer to 32,000 than to any other bit rate in the rules table.

Before the latest DSN upgrade there was no

Example 1. Spacecraft Initialization Table (partial)

```
#################################################################
# SPACECRAFT:   DSPSE
#
# Characteristics:      GSFC data type
#                       Single channel
#                       MCD coded
#                       NRZ-M to Bi0-L conversion
#################################################################
#
PGM     DSPSE
OFT1    DSPSE
#
# XBBM is internal form of the BBM OD
#
XBBM    UU
#
TBR1    128000  C
FSU1    S
FOM1    A
FLG1    P=2048   S=0
FBT1    IL=1  OL=1
FLT1    IL=2  OL=2
FSW1    32  EB90146F
APC1    D
FES1    E
MCV1    CCSDS
MLT1    3  3048
MCA1    MNR 256 2048
DIFD1   NM
RSU1    D
#
# XPRn replaces IGP and USP ODs
#
XPR1    D  D  D  D  D  D
#
PTO1    0  000000
MNO1    ACC=0.0  AGN=0.0  DOP=0.0  SCF=0.0  SNR=0.0  TBR=0.0
BBS     1, 9, 2, 10
LBW1    M  3
SER1    D
SLT1    -2.5
SNR1    0.0
SCF1    1700000.0
# Input symbol is NRZ-M
PCM1    NL
DSA1    E
```

Example 2. Rules Table

```
#
# SPACECRAFT:   DSPSE1
# DATE CREATED: 09/15/93 - First edition
#
TBR 125          TSO DSID=ED
TBR 250          TSO DSID=EE
TBR 500          TSO DSID=EF
TBR 1000         TSO DSID=F0
TBR 2000         TSO DSID=F1
TBR 8000         TSO DSID=F2
TBR 64000        TSO DSID=F3
TBR 128000       TSO DSID=F4
```

corresponding capability. Operators had to enter all ODs manually every time bit rate changed.

Example 2 is a rules table.

3.2.2 Tradeoffs in Rules Tables Design

Like SIT tables, rules tables use the same interface as ODs, making them easy from both a human and a machine perspective.

Conceptually, rules tables are almost part of SIT tables. Both are implemented from the same documents with invariant fields going to a SIT table while bit-rate dependent fields go to a rules table. Care is needed on the part of the table implementor to make sure all fields go to one or the other and to put only the needed fields in rules, as these will override operator-entered commands whenever bit-rate changes.

3.3 Formatter Tables

Format tables specify the packaging of data from an internal representation to the format required by the mission. (Formats currently supported include 1200- and 4800-bit asynchronous and frame-synchronous blocks as well as variable-length Standard Formatted Data Units.) In addition to data, formatted output generally incorporates a variety of information about processing: bit rate, sequence number, signal strength, Earth Receive Time, etc.

3.3.1 Formatting in the Bad Old Days

Before the recent DSN upgrade formatting was implemented directly in computer language, with a separate executable overlay for each supported mission. So every change to a formatter required an entire build and delivery, and the computer language implementation left open the possibility that typos could create apparently unrelated errors.

3.3.2 Format Tables

Format tables are much more complex than SIT or Rules tables. In essence they are almost a mini-language, but this language is focused only on the ability to format telemetry data.

The first part of each table (after a conventional comment header including name and change history) is a preamble that specifies whether the rest of the table will use 8-, 16-, or 32-bit words and whether word and bit counting will number from zero or one. This makes it easy to translate any document to a format table.

The rest of the table is divided into event blocks. Each event block specifies actions to be taken at a specific event:

- when the format table is first loaded
- when new information on upstream processing is received
- when new data is received
- when a new output block is started
- when an output block is completed

Within each event block is a series of table entries. These entries are executed in the order they are encountered. Generally it is preferable to order entries within each section by increasing address of destination within output data, but sometimes dependencies among fields make it necessary to vary this order.

The first column of each entry specifies the source, the second column specifies the destination, and the optional third column

specifies operations to be performed on the data.

Source and destination fields can be constants (source only) (numeric, restricted ASCII, or symbolic), fields from within the formatted data block (/[<start-word>].<start-bit>:[<word-length>].<bit-length>/), or named data store fields. Data stores correspond to formatter structures. There are structures associated with each input and output data block, with status information for display, with the formatting process, and with upstream processing information. A few fields are available for internal use when more than one operation is needed at a time.

Operations can be simple replacement, bitwise-or with current contents, floating point conversions, addition of a constant, table lookup, or if-style flow control.

Example 3 is a format table.

3.3.3 Formatter Implementation

For reasons of speed, operational software uses binary forms of format tables. The translation from text to binary format is normally performed when the delivery media is prepared.

Tables can also be modified and "binarized" (compiled) in the field if necessary using a standard text editor.

Inside the binarizer, the 'C' preprocessor is used in a way called Plastic List Manipulation to allow use of C structure field names to access those fields from within tables. The binarizer translates field names to structure offsets and lengths. Version checkwords make sure that the appropriate version of the binarizer was used on each file, so problems are not created when structures change.

3.3.4 Formatter Tradeoffs

Formatter design was essentially unconstrained by prior art, leaving many decisions to the implementer. The two major considerations were ease of implementation and ease of use. Ease of use seems best served by making the format of the tables as similar as possible to the format of the documents that specify them. In cases where document format could not work, ease of use is best served by similarity to familiar computer languages.

But with limited implementation effort, many decisions were made to favor easy one-pass interpretation. These include separating out event blocks instead of allowing pure ordering by address and placing the optional operation field last.

It is worth noting, however, that the binary file implementation leaves open the possibility that a "friendlier" binarizer could be written, producing the same binary format and therefor not impacting the formatter software at all.

It is also worth noting that user-friendliness is relative. Even the week that might be required for a new user to implement a new spacecraft is a great improvement over the previous "hard-coded" method.

4. Difficulties of Working with Tables

Adding table capability to any program increases its complexity and therefore its upfront costs.

Example 3. Format Table

```
;
;                Formatter Table for mission DSPSE
;

; The following fields tell the binarizer how to interpret the word.bit
; destination and length fields below.  They do not correspond to any fields
; of Out_Form.
WORDLENGTH=16     ; Number of bits in word for dest & length fields below.
FIRSTBIT=1        ; 1 or 0, if bits are counted starting from one or from zero
FIRSTWORD=1       ; 1 or 0, if words are counted starting from one or from zero

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

on(COND_NEWFORM); Tag meaning following operations are to take place when
                ; this file is first read in as a new format.
; source                  destination      operation (blank => simple copy)

FALSE                     of.ts_last_bit           ; Timestamp on last bit =
                                                   ; FALSE => timestamp on
                                                   ; first bit.

OPS67_FMT                 of.fmt_type              ; Format type: OPS-6-7,
                                                   ; OPS-6-8, or SFDU

NO_PAD                    of.pad_mode              ; Data pad mode: byte mode,
                                                   ; word mode, or no padding.

0                         of.pad_pat               ; Pattern for data padding.

ASYNCH                    of.in_per_out            ; Number of input frames
                                                   ; per output block.

144                       of.start_bit[0]          ; Bit of start of data field.
                                                   ; Bit 144 is first bit of word
                                                   ; 10.

4735                      of.end_bit[0]            ; Bit of end of data field.
                                                   ; Bit 4735 is the last bit of
                                                   ; word 296.

CHK_NO_REMOVE             of.chk_symb              ; Never remove RS check symbols

TRUE                      of.use_fsw               ; Do use synch word in output

0                         of.numfill               ; No (nonzero) data filling

600                       cfg.basetranlen          ; Number of bytes to transmit

0x627627                  /1.1:1.8/                ; sync code

0x11                      /3.12:5/                 ; Block Format Code.
; TCA always thinks it's real time.

0x46                      /5.1:8/                  ; Message type code
                                                   ; 26-meter throughput telemetry

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

on(COND_CONFIG)  ; Tag meaning following operations are to take place when
                 ; new configuration data (cfg.*) is received.

; source                  destination      operation (blank => simple copy)

cfg.src_code              /2.9:8/                  ; Source code
```

Example 3. Format Table (continued)

```
sap.scn                  /4.1:8/                ; Spacecraft Number

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

on(COND_IN)        ; Tag meaning following operations are to take place when
                   ; each input block is received.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

on(COND_BIT_1)     ; Tag meaning following operations are to take place when
                   ; a new output block has been started.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

on(COND_OUT)       ; Tag meaning following operations are to take place when
                   ; all data to be sent has been placed in the outgoing block.

; Hard-coded calculations will determine the out.out_dest used here
; from DEST specified above and TSO commands and the VCID.
out.out_dest             /3.1:8/                ; Destination code

out.out_dsid             /4.9:8/                ; Data Stream ID

sap.bsn_0                /5.9:8/                ; Message Block Count

; First three bits of word 6 are flags set to zero.  Ignored because all
; fields are zeroed.

out.fill_next            /6.4:13/               ; Number of telemetry bits in
                                                ; data field.

; First two bits of word 7 are flags set to zero.  Ignored because all
; fields are zeroed.

; following fields are timestamp of first bit in data field.

out.doy                  /7.3:9/                ; Day of year

out.msec_day             /7.12:1.11/            ; milliseconds of UTC

out.usec_msec            /9.7:10/               ; microseconds of millisec

; Telemetry data (and filler) words 10-296

sap.bsn_0                /297.1:16/             ; Block Serial Number

; Zero-filled CGF Error fields words 298-300
```

4.1 Documentation

When a new interface is invented for tables, as for format tables, it must be documented. The format table documentation, including examples, index, and descriptions for each data store field, runs to over 200 pages. The work of documenting an interface must be included in the cost of adding table capability. The size and complexity of the documentation seems to increase as tradeoffs are made to simplify the implementation of tables at the cost of more work in making tables.

4.2 Speed

The extra level of indirection introduced by tables carries with it a significant run-time computational expense. I estimate it at 10x for formatting header fields but no added expense in the move of actual data, averaging out to a 2x to 3x cost penalty. Current software and hardware can pay this penalty and still process data at the highest rate currently needed: 2.2 Mbits/sec. If higher rates are needed, special hardware may be needed.

5. Conclusion

DSN has now been operationally using format tables for nearly three years and SIT and rules tables for 1 year for all telemetry adaptation and in that time has successfully added over a dozen new missions without any required software changes. The only modification to the software has been to accommodate NIMBUS frame stripping, where most of the data is discarded because most of the instruments are no longer functioning. Even this could have been done with format tables, but would have been too slow.

In addition, turnaround time for for implementation of mission-specific changes from inception to installation has been reduced by a factor of four.

6. References

Deep Space Network Telemetry Group Controller Software Operator's Manual. TDA/DSN No. UG-DOT-5464-OP Rev. A. NASA Ident No. 23835. May 2, 1994.

Deep Space Network Telemetry Channel Assembly Software User's Guide. TDA/DSN No. UG-DOT-5464-OP Rev. B. NASA Ident No. 23835. July 16, 1993

7. Acknowledgements

The work described in this paper was accomplished by Telos Corporation under contract to the Jet Propulsion Laboratory, California Institute of Technology and sponsored by the National Aeronautics and Space Administration.

I would especially like to thank Michael Stoloff of JPL for his foresight and encouragement.

Within Telos, this effort benefitted from the support of Tom Soderstrom and the DSN telemetry operations expertise of Ron Holden.